


Awakening curiosity-Hardware education for Computer science students

Nevena Ackovska, Sashko Ristov

Related papers

[Download a PDF Pack](#) of the best related papers 



[The x86 Microprocessors \(Second Edition\)8086 to Pentium, Multicores, Atom and the 8051 M...](#)
lyla das

[First Pages](#)

SNZ INNOVATIONS

[Lyla B. Das, The x86 Microprocessors: 8086 to Pentium, Multicores, Atom and the 8051 Microcontrolle...](#)
Lyla B Das

Awakening curiosity - Hardware education for Computer science students

Sashko Ristov, Milosh Stolikj, Nevena Ackovska

Ss. Cyril and Methodius University / Faculty of Natural Sciences and Math. - Institute of Informatics, Skopje, Macedonia
e-mail: {sasko, milos, nevena}@ii.edu.mk

Abstract — This paper describes a new systematic approach and methodology for teaching hardware based courses to computer science (CS) students. It is a combination of various approaches with a profound goal to provoke deeper comprehension in various topics in microprocessors and microcontrollers details. The computer student programming skills and knowledge (generally high level programming languages and algorithms) is used to program and control processes with microcontrollers. The methodology evolves around three steps: using visual simulators, incrementally weighted exercises, from easiest to hardest, including supplemental points, and finally working on real hardware controllers. The proposed teaching approaches were developed for the course “Microprocessors and Microcontrollers”, but can be applied to every hardware based course on computer science students.

I. INTRODUCTION

Computer science and CE courses in general tend to adopt the usage of e-learning platforms. This approach is appropriate for all software oriented courses and those ones that do not need considerable instructor control. All software courses have huge benefit from the new way of learning and teaching, but, as it is shown in many cases, it is not the best solution for courses which are not so native with the appropriate science. Hardware courses existing in Computer Science (CS) or Computer Engineering (CE) curriculums fall in that category. To this end, computer architecture and organization is an important area in undergraduate CS curricula and it is acknowledged as a substantial part of the body of knowledge in [1].

However, computer science has been bloated by new subject areas. That new burden of knowledge stimulates a reorientation favoring courses about higher abstraction notions (e.g. web-oriented programming), while courses about low-level details (e.g. computer architecture) are abandoned or severely minimized. The problem that inherently appears is that high-level programming does not provide a clear understanding of what a computer really does when executing a program. So, students do not want to know how the computer works, but only that they can execute their software solutions on it. Even more, reorientation from

computing to software programming inevitably provokes students to feel that hardware oriented courses are simply a burden that they have to put up with in order to gain their grade and pass the exam.

Teaching students how hardware devices work and how they can be employed in their designs is often a very difficult process. For the educator, hardware based courses are a challenge and seeking new teaching methodologies is a continuing process. There are many different methodologies and approaches in teaching computer science students about hardware from different points of view. For instance, [2] presents an approach for getting the students hands-on experience on the software (operating system) and hardware interaction. In [3] the focus is on embedded software and systems as part of teaching and learning. Also, [4] presents the experiences while using the simulation in an introductory microcontroller class. In [6] we observe common methods of class room teaching and experimental teaching, with the emphasis on cultivating the problem-analysis and the problem solving abilities of the students on computer hardware curriculums.

A. Common hardware courses

We, at the Institute of Informatics, offer three related courses that cover issues of computer architecture and organization. These courses are named “Computer Architecture”, “Microprocessors and Microcontrollers” and “Modern Computer Systems”. In this paper, our interest is the course Microprocessors and Microcontrollers. Its main objective is for the students to obtain a clear understanding of issues such as low-level hardware interfacing, handling of interrupts, communication between processor, memory, bus and peripheral devices through learning the basics of processors and its instruction set, as well as embedded systems through learning microcontrollers. Hands-on experience is considered very important. This paper presents our improvements to the course curricula and summarizes the new teaching methodology.

In the following section, we describe the course syllabus and difficulties that students have in understanding low-level computing issues. In the third section we describe our rearranged systematic approach to teaching scope and

methodology for making microcontrollers and their functions and usage to computer science students. In the forth section we present the results and improvements obtained by integrating the changes in the course curricula. At the end we present the future expectations and the conclusions about this methodology applied into our course.

II. BACKGROUND

A. About the course

The course “Microprocessors and Microcontrollers” is an obligatory course, primarily intended for computer science students in their third or fourth year of studies. Prerequisites for enrolling in the course are previously completed courses in Computer Architecture and Operating Systems.

The course classroom teaching is organized in three parts: theoretical lectures with 2 classes per week, theoretical exercises with 1 class per week and practical exercises with 2 classes per week in laboratory. Lectures and theoretical exercises are organized in larger groups, while practical exercises are carried out in computer laboratories in groups of up to 20 students, with each student working on its own station.

The teaching material is divided in two parts. The first part covers the internal architecture and instruction set of x86 microprocessors, the interrupt handling system, BIOS and system routines. The second one focuses on various types of microcontrollers, analyzing their organization, instruction set and capabilities compared to x86 microprocessors, as well as peripheral systems, embedded systems etc.

Lectures and theoretical exercises for the first part of the course are based on [7] and various source code libraries. In previous years, programming was done using Microsoft Macro Assembler (MASM) version 6.11, with Programmers Work Bench (PWB) as an integrated development environment [8]. The material for the second part of the course depends on the microcontroller that was studied.

Grading in the course is divided into four categories - student activity, projects and either two midterms or one exam. Student activity is followed during the entire length of the course in the form of obligatory laboratory exercises, which are intended to be solved during classes. Two projects are handed out, one for programming in x86 assembly language, and the other one for programming a microcontroller. These are more complex problems, often requiring additional research and learning area not covered in the course. Finally, midterms, or later exams, are conducted for testing both theoretical and practical knowledge.

B. What and why is it hard to learn?

The course had a “bad” reputation of being abstract, boring and very hard to comprehend. Some of the issues were subjective and student-specific, while many others were completely in place. Most common objections to the course were:

- Inappropriate programming and simulation environment. Students had problems installing and using PWB, which caused aversion to the presented material and exercises.
- Disjointed material between lectures, theoretical and practical exercises. Students could not transfer knowledge gained from either lectures or theoretical exercises to practical exercises. This made them think they were studying two or three courses in one.
- No “real world” application. Without having direct hardware interaction, students learning becomes abstract, which leads to their displeasure and to the main question: Why we are learning this, and how and where shall I use it?

III. SYSTEMATIC APPROACH

As undergraduate educators, one of the most important considerations is to present the subjects of the course in a way that it will provoke student’s interest and initiate self-driven actions. These objectives become even more evident and challenging in courses such as the “Microprocessors and Microcontrollers” for CS students.

A. Analyzes

We analyzed the hardware courses during the period of several years, and noticed that a lot of excellent software contestant students had huge problems in hardware courses. There were many students, who were national software contestants and winners, as well as CodeFu Java contest winners, that had problems with passing the hardware based courses, or were getting only lower grades.

Our idea was to construct a “hardware” course that is more similar to “software” one. An approach like this enabled an easier starting point for our students since they were already familiar with high level programming languages like C++.

B. What did we change in the course?

Most changes in the course originated from laboratory exercises. Lab exercises handouts were in the form of tutorials. First they briefly cover the material given during theoretical lectures, including code samples how previously taught principles can be implemented. Afterwards, relatively simple assignments are given, requiring students to apply that knowledge for certain tasks. This part of the exercises was graded and had influence towards the final outcome of the course.

C. Make the course more visual and closer to software

Initially we switched from using PWB for x86 assembler to modern visual simulators like emu8086 [9], where students can watch parameter values, i.e. registers, memory locations, as well as executing programs step by step in real time. They can use their CS knowledge of debugging for easy troubleshooting. Modifications in the handout material

were oriented towards matching the theoretical lectures. We also left the old “hardware” oriented exercises which taught the students working with graphics, operating systems, files etc, but introduced more software exercises such as arrays and matrix, strings, procedures, macros.

For the second part of the course we decided to switch from the previously used Intel 8051 architecture and PIC 8259A interrupt controller to some other implementation.

As a new platform we selected the microcontroller PIC16F887 [10], used on the EasyPIC6 Development System (Fig. 1) made by MikroElektronika [11].

D. The PIC 16F887 Microcontroller

There is a variety of available microcontrollers today, like the 80XX family, Arudino, PIC etc. We decided to use the PIC platform on the basis of the following premises:

- Large number of implementations, including free and commercial.
- The availability of the embedded software providing a direct connection with PC and the availability of Flash memory. This allows easy programming and re-programming of the memory as well as software adjustment and debugging.
- A variety of controllers embedded in the microcontroller, which allows creating a lot of real world examples.
- Free circulation of the IDE which allows reading, loading, adjustment and performance of software in a real-time mode and in a simulation mode.
- Free circulation of a limited, but fully function version of the MikroC compiler for binaries up to 2KB.
- Accompanying literature about the microcontroller and its programming [5].
- The availability of a lot of complete project examples with the microcontroller.



Figure 1. EasyPIC 6 Development System

In specific, the PIC 16F887 microcontroller has RISC architecture and only 35 instructions. It can operate on frequencies up to 20 MHz, using an internal or external oscillator. Connection to peripherals is made through 35 programmable input/output pins. Three types of memory are available – 8KB ROM memory, 256 bytes EEPROM memory and 368 bytes RAM memory. In combination with the EasyPIC6 Development System the powerful In-circuit Debugger enables real time execution and debugging of code directly on the microcontroller, with options for step by step execution, monitoring of memory, registers etc.

The company MikroElektronika offers variety of hardware (from PIC series), as well as a lot of software support. This enabled us to include a lot of sensors, actuators, as well as small control systems as part of exercises and exams.

E. The transformation

Computer science students know how to solve issues when writing software, whether it is for the web or desktop applications. The main scope of our approach is to assure students that their programming skills can be used for controlling hardware as well. If they can see results of their work immediately, we presume that they will be satisfied and will begin to appreciate the course itself.

The first practical exercises were made in that direction. For instance, students were shown how traffic lights on crossroads can be programmed. Then, with more intelligent algorithms and sensors, they proposed how traffic lights can become more intelligent, for handling congestions, controlling accidents etc.

For the second part of the course we relied on the microC programming language for programming the PIC16F887 microcontroller. Combined with the excellent high level application program interfaces in the MikroElektronika microC PRO compiler [12], students could use many additional hardware components such as alphanumeric LCD displays, ports for serial communication, EEPROM memory, with only a few lines of code.

Since we had several boards available for experimenting, we offered optional projects to students which involved working with new hardware components which were not available during classes. Each project was based on a unique component: Graphic LCD, Touch Panel, Ethernet board, Digital thermometer, Motion Sensor etc, for which students had to work out by themselves how to control and use them. They often had to fight memory and processing limitations of the microcontroller, compared to the high performance of the x86 microprocessors.

F. Teaching methodology

To achieve the scope, we propose the following methodology for teaching microprocessors and microcontrollers to CS students:

- Use easy to use visual simulators at the beginning of course to work on the easy and answered exercises. CS students do not even think of hardware and how hardware components work. They have problems in understanding and visualizing the micro hardware architecture. Starting directly with hardware exercises will turn away student attention off the course.
- Start with incrementally weighted exercises, as well as opportunities for bonus scores. It enables easier grouping of students based on knowledge and interest. Advanced students want to have higher grades and they must be allowed and encouraged to work on harder tasks. On the other hand, many students just want to learn only what they have to to pass the exam. Bonus scores keep advanced students challenged and focused through the entire course.
- Incorporate real life problems in exercises using physical hardware, not just simulators. After students learn the basic hardware components, as well as their processing and programming routines, they must experience hands-on activities.
- Give an option for non-obligatory projects, which can help students in their grade. The projects were chosen from the student real life, such as: Reading temperature or motion sensor and send the information on server through ethernet, basketball realtime scoreboard and time display, Calculator, then well known games like Millioner, Sokoban, Minesweeper (Fig. 2), Frogger, X/O, Snake, Tetris, Formula, etc.

As practice shows, they are only selected by the best students enrolled in the course, and almost certainly produce impressive results.

IV. RESULTS

The introduction of this course, tools, and platforms has been underway for several years, but we can already observe

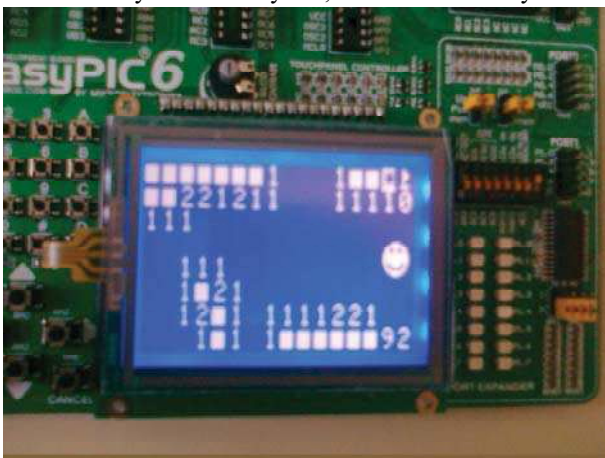


Figure 2. Practical projects example: Minesweeper

significant positive changes in our students after only one year.

A. Significant changes: practical projects

For instance, student feedback on the practical projects (defined as additional) has been extremely positive. More than 75% of the students submitted their projects on time, compared to 68% from last year. On top of that, 14% of the students chose an additional project, compared to last year's 1%. Most of these students had already passed the exam with the highest grade and did not need the additional points. Also, many of the students are not living near the Institute of Informatics, so they had to travel during the holiday season just for working on the practical projects (Fig. 3). This could not be the case the last year, for example. We can only conclude that students appreciated the integrative nature of the projects, where computing was interlinked with traditional engineering.

B. Better results in laboratory exercises

Improvements are evident in the submission of laboratory exercises as well. We created totally new laboratory exercises for Intel 8086, as well as for PIC16F887. The results were striking, especially those on hardware. More than 75% (average) of students won a bonus points for the last four laboratory exercises. Last year, only 18.5% of students won maximum points for the last four exercises.

C. Better final grades

Another improvement is evident in the grades distribution, from 6 (smallest grade to pass) to 10 (highest grade).

In the previous year, only 11.5% of the students (15 of 130) passed the course through midterms. This year, the number of passed student has almost doubled to 22.6% (26 of 115). It is significant to note that four of these students have been already enrolled in the previous year course.



Figure 3. Student project on EasyPIC 6 Development System

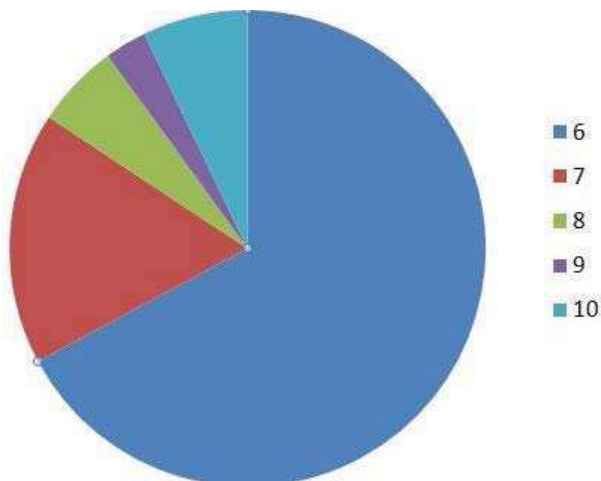


Figure 4. Grade distribution: Previous year

Fig. 4 shows the grade distribution in the previous year and Fig. 5 shows this year grade distribution with implementation of the new methodology. In the previous year, more than 2/3 of passed students passed the course with a minimal grade 6. Also, only 10% of passed students got the highest grades 9 or 10. This year, only 32% of passed students passed the course with a minimal grade 6. Fascinating 30% get the highest grades 9 or 10.

D. Exceptional additional student projects

15 students (14%) applied for and completed additional practical projects, although we defined only 10 additional projects in the beginning. Students faced a lot of microcontrollers constraints (deficiency of RAM and ROM memory, pins and Cyrillic font), but effectively solved the problems using their software skills and algorithms. We are surprised by the promptness of students' solutions submission.

The best 7 projects were presented on the CIIT conference [13], 5 of which were presented on hardware session and others 2 (the best ones) on the student session. The company that donated the equipment was pleasantly

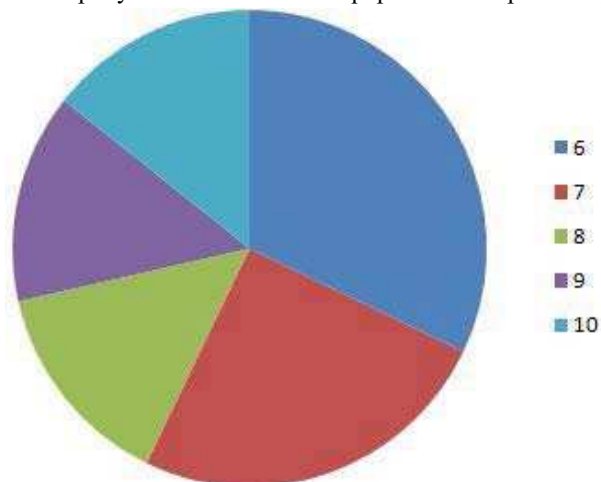


Figure 5. Grade distribution: With new methodology

surprised of the projects difficulty and the students' effectiveness to achieve a solution, as well as solutions themselves, so they awarded 4 best projects, instead of planned 1. All additional projects were evaluated with more than 10 points, and the best projects were evaluated with 30 points. Fig. 6 shows the grade distribution of all projects (obligatory and additional).

V. FUTURE DEVELOPMENT

The course in its current form is still young and subject to changes. Improvements in the curricula are always welcome and broadly accepted.

The growing student interest in the course led to the development of a specialized laboratory for Microchips at the Institute of Informatics. The laboratory is open to all students interested in researching and experiments with microprocessors and microcontrollers, or other embedded system. It is intended to complement the existing Robotics and Intelligent systems laboratory, supplied with several manipulative, walking and mobile robots. The goal is to increase not only CS student interest about microcontrollers and microcomputers, but also the other students for which this course is optional.

Other issue we must improve is the percentage of (non)passed students. The percent of unsolved projects was basically the same in the past two years (more than 30%), which automatically means that those students didn't pass the course.

VI. CONCLUSION

The changes implemented in the course "Microprocessors and Microcontrollers" managed to awake the students' curiosity about hardware and mapped their software knowledge into hardware oriented problems. This took a lot of effort, both from the instructors and from the students.

There are significant improvements in the grade

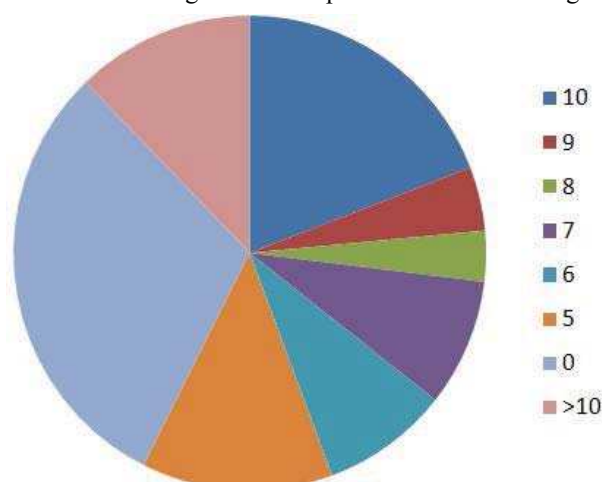


Figure 6. Project grade distribution:

distribution, as well as in the students' interests in working on hardware equipment. This resulted in significant number of completed hardware projects, although they were not obligatory in order to pass the course.

The growing student interest in the course led to the development of a specialized laboratory for Microcontrollers at the Institute. The laboratory is open to all students interested in researching and experiments with microprocessors and microcontrollers, or other embedded system. It is intended to complement the existing Robotics laboratory, supplied with several manipulative, walking and mobile robots. The goal is to increase not only CS student interest about microcontrollers, but also the other students which this course is optional.

VII. ACKNOWLEDGMENTS

The authors would like to thank the company Loging Electronics for their generous donation of the equipment that was used in the laboratory exercises and additional student projects.

REFERENCES

- [1] The joint task force for computing curricula of The Association for Computing Machinery, association for computing machinery and IEEE computer society and IEEE/ACM (2001). Computing Curricula 2005. The Overview Report. 2005.
- [2] Dimitris Kehagias, Michael Grivas: "Software-oriented approaches for teaching computer architecture to computer science students". Journal of Communication and Computer, 2009, ISSN 1548-7709, USA
- [3] Sztipanovits J., Biswas G., Frampton K., Gokhale A., Howard L., Karsai G., Koo T.J., Koutsoukos X., Schmiat C. "Introducing embedded software and systems education and advanced learning technology in an engineering curriculum". ACM Transactions on Embedded Computing Systems, 2005, 4(3): 549-568.
- [4] Carl E. Wick "Teaching Embedded Computer Systems with a Windows-Based Simulator", 1996, IEEE Catalog Number: 96CH35946, ISBN Number 0-7803-3720-4
- [5] Milan Verle "PIC Microcontrollers - Programming in C". MikroElektronika; 1st edition (2009), ISBN-13: 978-86-84417-17-8
- [6] Li Da, "Computer Hardware Curriculums, Curriculum Contents and Teaching Methods". Proceedings of 2009 4th International Conference on Computer Science & Education.
- [7] Randall Hyde, "The Art of Assembly Language", ISBN 1-886411-97-2
- [8] Microsoft Macro Assembler, <http://www.masm32.com/history.htm>; Retrieved on 02.02.2011
- [9] Emu 8086, <http://www.emu8086.com>; Retrieved on 02.02.2011
- [10] Microchip PIC 16F887, <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en026561>; Retrieved on 02.02.2011
- [11] Mikroelektronika, <http://www.mikroe.com>; Retrieved on 02.02.2011
- [12] microC PRO for PIC, <http://www.mikroe.com/eng/products/view/7/mikroc-pro-for-pic/>, Retrieved on 02.02.2011
- [13] The Eighth Conference for Informatics and Information Technology with International Participation CIIT2011, <http://www.ii.edu.mk/ciit/>